Preprint
UCRL-JC-144254

Software Authentication

J.K. Wolford, B.D. Geelhood, V.A. Hamilton, J. Ingrahan, D.W. MacArthur, D.J. Mitchell, J.A. Mullens, P.E. Vanier, G.K. White, and R. Whiteson

This article was submitted to
42nd Institute of Nuclear Materials Management Annual Meeting,
Indian Wells, CA July 15 – 19, 2001

June 21, 2001

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

Approved for public release; further dissemination unlimited

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

This report has been reproduced
directly from the best available copy.

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information
P.O. Box 62, Oak Ridge, TN 37831
Prices available from (423) 576-8401
http://apollo.osti.gov/bridge/

Available to the public from the
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd.,
Springfield, VA 22161
http://www.ntis.gov/

OR

Lawrence Livermore National Laboratory
Technical Information Department's Digital Library
http://www.llnl.gov/tid/Library.html

SOFTWARE AUTHENTICATION

J.K. Wolford, Jr. (LLNL), B.D. Geelhood (PNNL), V.A. Hamilton (SNL), J. Ingraham (DynCorp), D.W. MacArthur (LANL), D.J. Mitchell (SNL), J.A. Mullens (ORNL), P.E. Vanier (BNL), G.K. White (LLNL), R. Whiteson (LANL)
Members of The Authentication Taskforce, Software Working Group

Lawrence Livermore National Laboratory

P.O. Box 808, Livermore, California 94550, USA, (925) 422-7236, wolford@llnl.gov

ABSTRACT

The effort to define guidance for authentication of software for arms control and nuclear material transparency measurements draws on a variety of disciplines and has involved synthesizing established criteria and practices with newer methods. Challenges include the need to protect classified information that the software manipulates as well as deal with the rapid pace of innovation in the technology of nuclear material monitoring. The resulting guidance will shape the design of future systems and inform the process of authentication of instruments now being developed. This paper explores the technical issues underlying the guidance and presents its major tenets.

INTRODUCTION

Authentication as practiced in a nuclear materials measurement program establishes trust between the monitoring party to an arms control or transparency agreement (monitor), and the holder of the nuclear material (host). The Joint DoD/DOE Authentication Taskforce[†] defines authentication as "the process through which the monitoring party gains appropriate confidence that the information reported by a monitoring system accurately reflects the true state of the monitored item." Determining what level of confidence is appropriate for a given agreement is the first step in planning for authentication. Authentication is made more difficult when there is a need to protect classified information and consequently the host supplies and perhaps even manufactures the monitoring system. Also, the increasingly common scenario of unattended monitoring complicates the authentication task still further.

Even within the context of a complete measurement system, the problem of authenticating software is broad and difficult. It spans the full range of considerations from high-level design criteria to the details of how code is executed by specific processors. It consists of two main parts:

(a) Demonstration that software operates as designed.

(b) Demonstration that software will not operate in a faulty or otherwise undesired manner.

[†] A group composed of representatives of the U.S. Department of Defense, the U.S. Department of Energy, and their contractors.

While the demonstration referred to in (a), that software operates precisely as designed, requires a rare degree of exactitude, the problem is nonetheless tractable. A rigorous demonstration that software is not faulty (b) would require both knowledge of all operating modes of a system and tests subjecting the system to the full range of input combinations it could possibly encounter. For systems of any complexity, this is not feasible. Formal methods exist for addressing the problem, but again, for all but the simplest systems, complete assurance is unattainable and remains a research area in computer science. Given that *absolute* confidence is impossible, the present approach focuses instead on gaining *appropriate* confidence as the definition of authentication requires.

Scope

The Authentication Taskforce Software Working Group has defined software design guidelines to be used by engineers and scientists in creating new measurement systems. These guidelines apply to all software to be used in measurement systems employed for arms control and transparency agreements. They encompass any software present in a measurement system, including data acquisition and analysis application software, information barrier software, computer operating system software, computer basic input/output system (BIOS) software, and firmware for programmable logic devices. The guidelines also extend to any software written for use in support of the on-site authentication process.

Motivation

Authentication is necessary because the monitor must be able to trust the results of measurements carried out under terms of agreements for arms control or transparency of nuclear material. Authentication is especially necessary when the measurement equipment, including software, is supplied by the host. (Host supply of measurement equipment is a reasonable assumption in any agreement involving weapon grade material, since the requirement to protect information considered classified or sensitive by the host country predominates.) This paper accepts the broad definition of *host supply* as adopted by the Joint DoD/DOE Information Barriers Working Group (IBWG.)[1] This definition permits equipment that the host obtains from foreign sources to be considered host supplied provided the host had sole custody of this equipment at any time prior to acceptance. However accepting host custody of any duration accentuates the need to authenticate all components, including any commercial products.

To reiterate, the Authentication Taskforce assumed that the host is primarily concerned with protecting its classified data and the process of certifying that protection, while the monitor is primarily concerned with system integrity and the authentication process. These premises apply to both bilateral and multilateral agreements. In general, the monitoring entity has high interest in the authenticity of the attribute results and indirect interest in whether or not the system protects the host's classified information, inasmuch as future agreements may contain reciprocal requirements. Similarly, any host interest in authentication anticipates future agreements where roles may switch. All parties value system reliability to a moderate degree, which is in part affected by the content of the monitoring agreement.

Two Main Forms of Guidance
Authentication guidance provided by the monitor to the host has two components. It offers initial design guidance for foreign-supplied monitoring systems, consisting of requirements and specifications that support sufficiently thorough and straightforward authentication. It also provides a prescription for the process of authentication. These subjects are treated separately in some detail below.

Complicating Effect of Information Barrier
The information barrier, where present, heightens the need for well-planned authentication. An information barrier is formally defined as "technology and procedures that prevent the release of host country classified information to a monitoring party during a joint inspection of a sensitive item, while promoting assurance of an accurate assessment of host country declarations regarding the item." Inherent in the notion of information barrier is the need to restore some of the assurance of authentic functioning that is lost by restricting the monitor's access to the measurement data. The information barrier places deliberate limits on the amount and type of information that a measurement system may reveal, adding to the importance of granting the monitoring party ample opportunity to inspect and test the system, including the software, before trusting its results. The loss of access to the measurement data hidden by the information barrier can be mitigated by software design that emphasizes simplicity and clarity and helps to build confidence in the proper operation of the system through other means, such as fully open testing on unclassified measurement items.

The Authentication Taskforce addressed system integrity and authentication but did not treat protection of classified information explicitly. Thus its guidance omitted any reiteration of the design basis for information barriers. The guidance assumes that the host will stipulate adequate requirements governing information protection concerning its material. However, since corresponding systems used for reciprocal agreements must address the current host/former monitor's confidentiality requirements, authentication guidance must be informed by domestic certification requirements.

Role of Documentation
The success of authentication depends on the host's willingness to be open. In general, this means that the supplier of the software (host) must provide sufficient documentation to enable the authenticating party (monitor) to understand the operation of the software to an adequate degree of detail. This establishes both an initial level of confidence and a knowledge base for subsequent comparison. Since nearly every software component is able to alter the outcome of a measurement, the host must provide enough breadth and depth of design information to enable the monitor to determine whether and where the software offers an opportunity for the host to alter the measurement results. Consistent application of this requirement is key since overall system confidence reduces to the level of confidence in the least understood and examined component. Software openness applies to source code and executable code as well as to the detailed procedure followed to derive the latter from the former. The importance of openness persists into the operations and maintenance phase, where the monitor has an enduring need to determine that the operating software remains identical to the software authenticated prior to acceptance.

SOFTWARE DESIGN FOR AUTHENTICATION

Relationship of Authentication to Security Certification and Overall Reliability

Design criteria affecting authentication overlap strongly with general principles of good software design. They also complement the requirement to protect classified information. The diagram below depicts this relationship.

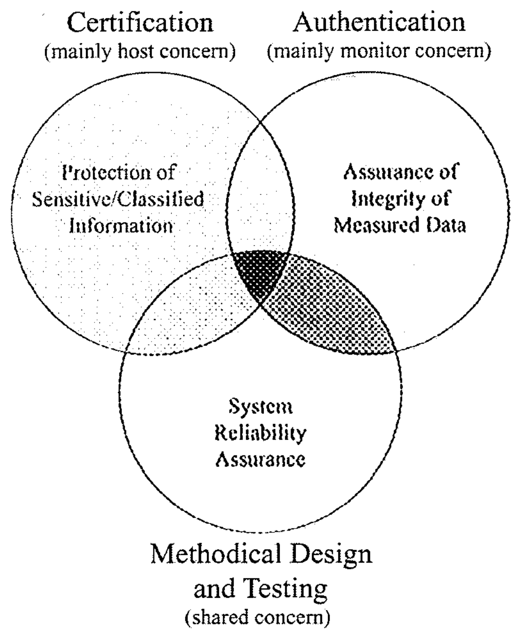

Figure 1. Venn diagram showing the three main considerations of software design for monitoring systems and their interrelationships. Both host and monitor have an interest in software reliability.

Both the host and the monitoring party have an interest in creating reliable and maintainable software. While software maintenance per se will probably not extend beyond the correction of flaws discovered during operation, the design practices that make software more maintainable will, in most cases, also make it easier to authenticate. As the diagram implies, portions of both the authentication and security certification goals are accomplished "for free" through the design and implementation of a reliable system.

Design Guidelines

The following table summarizes the high-level design guidance provided by the Software Working Group, along with its rationale. These six guidelines form a basis from which design requirements may be derived. Where authentication design features and activities are subject to negotiation, knowing the rationale behind the considerations may be useful.

Table 1. High Level Software Design Guidelines

Guideline	Rationale
Agree on software to be used.	Knowledge on the monitor's part of software to be employed by the host is the first building block of confidence and helps frame the authentication problem.

Develop software according to agreed-upon international standards.	Software standards promote robust, reliable, readable, and correct software. International standards carry no national bias.
Conduct reviews throughout the software life cycle and especially the design phase.	Authentication must continue throughout the software life cycle to maintain confidence..
Reserve the opportunity to inspect and test the software prior to acceptance to gain an appropriate level of assurance.	On-site authentication activities are insufficient. The monitor must be afforded the opportunity to thoroughly analyze the software at a facility under the monitor's control.
Design the system to support occasional inspections to ensure that equipment in operation is identical to what was accepted.	The monitor must have assurance that the operational software identically matches the accepted software.
Create a means to agree on any changes to the software taking place after system acceptance.	Monitor must re-authenticate the system, or portions thereof, after even minor software changes. Changes to software may include those requested by the monitor to promote authentication.

Role of Software Engineering Techniques

Software design in support of authentication, or of any goal, benefits from a well-defined software engineering process. The U.S. DoD has sponsored the Software Engineering Institute at Carnegie Mellon University to create standards and models for the development of secure and reliable software. Chief among them, the Capability Maturity Model, gives high-level guidance for the entire software life cycle and is often used in process evaluation.[2] The appropriate software engineering process for a regime-specific system should be determined by considering the risks to the system as well as the "capability maturity" of the software development organization.

Authentication-Specific Design Reviews

Prior to the start of the software design process, the monitoring party should articulate an authentication policy specific to the agreement that the measurements are to support. This policy should express the monitor's authentication goals in a format that may be shared with the host's software design organization. This policy may include formal statements of authentication-oriented design criteria, or the criteria may form the basis of a separate document. Either way, the host's design documentation should refer to the design criteria, and eventual testing of the software should comprehensively cover those criteria. The following table summarizes in chronological order the suggested design phase documents that create the basis for authentication.

Table 2. Suggested Design Phase Authentication Documents

Document	Purpose
Authentication Policy	High-level statement by monitor of goals and expectations for authentication activity
Authentication-Specific Requirements	System design requirements derived from the authentication policy and compiled by the monitor in collaboration with the host
Authentication Test Plan	Description of approaches and methods for testing the system against its authentication requirements
Design Review Summaries	Reports recording the conduct of design reviews, compiled in collaboration

THE SOFTWARE AUTHENTICATION PROCESS

The following table gives a broad summary of system authentication activities relevant to software. It is divided into columns according to a simplified software lifecycle. The remainder of this section consists of elaboration of those activities.

Table 3. Software Authentication Activity by Lifecycle Phase

Simplified Lifecycle Phase	Design	Fabrication / Implementation	Installation / Acceptance	Operation / Maintenance
Software Authentication Activity	Reviews	Unit/module and integration test witnessing	Random selection of full system, including software media. System level testing	Random selection of components. On-site checks.

Design Reviews (Design Verification)

The proposed method for verification of the software design consists of a desktop review of the design documents to ensure that the software functional requirements will be met. Among other things, this process determines whether the overall system requirements will be met if the software works as designed.

Preliminary and Critical Design Reviews, such as those routinely performed for development of critical software in the U.S., could serve as a model. (*Critical* is usually defined as software whose failure would have costly or life-threatening consequences.) This involves, among other things, answering the following questions:

- Basic Functioning: Will the acquisition and analysis techniques underlying the design provide the required attribute or template results?
- Performance: Will the acquisition and analysis software provide results of sufficient accuracy to meet overall measurement system requirements (e.g., acceptable false alarm rates, etc.)?
- Connection to Procedures: Do the operational requirements of the system cause any procedural problems, e.g., would the system require operators to perform actions that violate facility procedures?
- Exception Handling: Does the design account for known off-normal conditions? The design documents should include a complete list of handled exceptions and a description of the action taken by the software in each case.
- Diagnostics: Within the constraints imposed by the information barrier, does the design provide a means for registering diagnostic information helpful in detecting faulty modules?
- Inspectability: Does the design limit the complexity of the overall system? Is the design simple enough so that unwanted modifications would be difficult to conceal?

Random Selection

A cornerstone of the authentication process is procuring from the host a representative system to inspect in detail. One proposed method for assuring this identity for entire systems requires the host to prepare more than the number of measurement systems needed for implementing the agreement. From this set of duplicate systems the monitor selects at random one or more of these systems to take possession of and one or more systems to serve as operational units in the host facility. This random selection method assumes that the host would supply the software as part of the measurement system at the time of the monitor's choice. Assurance will further depend on technical measures applied to the chosen operational units to prevent tampering or substitution either prior to or during use, e.g., appropriate seals and/or other tamper indicating devices.

Random selection could be costly for expensive measurement systems; however, the concept could be applied usefully to less expensive subsystems. Moreover, component-level random selection could be more easily applied, and the inspection be made more focused, if the subsystems involved were modular and easily separable. Since software-bearing modules can be made separable from the rest of the system by design, a modified form of random selection involving only the software-bearing elements (e.g., computers, read-only-memory components, or other media) might be performed. Random selection of software-bearing subsystems *only* might offer a cost-effective alternative to routine evaluation of an entire system. Procedures to preserve Continuity of Knowledge for these separable components and their replacements should be defined beforehand.

Software Testing by Inspection (Design Validation)
Design validation involves evaluating the software by inspecting the source code and comparing its content to design documents.[†] It addresses the question of whether the software does what it was designed to do, whether it does it correctly, and whether it does nothing beyond what it was designed to do. To facilitate this human time–intensive activity, the "amount" of software should be minimized and its design made clear. (Beyond these measures, developing software jointly with contributions from both host and monitor would support planning in advance for authentication, particularly if the design documents were developed jointly.) The subtasks involved include:

- Validation that software is adequately specified. Does the developer-supplied user documentation adequately explain the presence of all software features?
- Validation that the software's platform requirements have been met. The hardware platform should be completely specified, e.g., exact processor and revision if any, BIOS version if present, any custom chips used in the system, etc.
- Validation that execution paths indicated in the source code will be interrogated during testing. (Test coverage analysis is addressed in the software test section.)

[†] A machine-based validation exercise might well be done using a debugger on the generated executable (disassembled executable) in addition to inspection of the source code. All other things being equal, the developer's implementation should facilitate this analysis. This is one reason to require source for libraries, etc. Also, the chosen development environment should support analysis tools such as debuggers and code analyzers or equivalents.

- Validation that source code has been developed according to accepted practices. Examples of such practices are contained in international standards and U.S. software security documents.[3,4]
- Validation that source code developed or acquired for use in the measurement system may be built into executable images that are identical to ones supplied by the developer.

Software Testing by Execution

Software testing by execution (also called dynamic testing) evaluates software against the requirements that governed its development. Test cases tied to software requirements are developed and exercised jointly by host and monitor. (Other tests may be applied by the monitor at its home facility using a randomly selected system.) Testing methods include inspection and execution. The depth of testing applied depends on the assurance level sought. In keeping with testing practices advocated by U.S. and international standards organizations, test cases should be compiled for testing at several levels including unit/module, integration, and full system tests, and should provide for regression testing as revisions to the software become necessary. Test cases should cover software features specific to the type of measurements performed as well as basic functionality (I/O, etc.). Regression testing should apply selected subsets of the tests. They should include but would not be limited to:

- Tests of any calibration, background, or other measurement control steps performed prior to actual material assays.
- Tests that take account of the statistical variations inherent in the raw measurement data.
- Threshold or near-threshold testing to verify an expected statistical distribution of results.
- Stress tests that deliberately introduce problems both within and between modules to verify adequate software robustness against failure. Stress testing seeks to disclose the limits of software reliability.

Post-Installation Authentication

Another main element of authentication is securing trust that the copy of the software presented to the authenticating authority is identical to what the fielded measurement system will contain. Checks to ensure that the software operating the measurement system is identical to a "golden copy" should precede each measurement campaign. The golden copy is established at the moment the software is accepted for use by both sides. That acceptance, presumably, will come as a result of a detailed initial authentication process by the monitoring entity. Additional confidence can be gained by an on-the-spot, bit-by-bit comparison of installed software against the golden copy. This comparison would naturally need to be performed using equipment that the monitor controls; otherwise the problem of trust merely shifts from the tested software to the comparison equipment.

COSTS AND BENEFITS OF AUTHENTICATION

Developers should recognize that various levels of authentication may be applied depending on an assessment of the costs and benefits involved. The goal of authentication at any lifecycle stage is to gain added assurance that the results produced by the measurement system are accurate and credible. Adequacy of assurance depends on the goal of the measurement regime. Generally speaking, the cost of authentication scales with the depth to which it is performed and the degree of assurance thereby attained. Full assurance may only be approached asymptotically, and the expending of considerable additional resources may only purchase incremental assurance gains, once a basic level has already been achieved. Features common in modern programming practice may also complicate authentication, making scrutiny of the initial design a worthy investment in critical systems.

Evaluation Assurance Levels

The process of gaining authentication assurance is analogous to gaining security assurance for systems trusted with sensitive information, and certain methods of security evaluation would also apply to the authentication process. An international standard on software security[5] defines seven evaluation assurance levels that grant increasing degrees of assurance. They are, in order of increasing probity:

- EAL1 - Functional testing; used when risks are not considered serious, and may be accomplished without access to the developer or any but the most basic design materials. Also called "black box" testing.
- EAL2 - Structural testing; requires contact with the developer and the sharing of design information and unit test results. Not substantially more costly than EAL1.
- EAL3 - Methodical testing and checking; places some attention at the design stage on assurance goals but focuses mainly on independent "gray box" testing. Access to developer still somewhat limited.
- EAL4 - Methodical design, testing, and review; more emphasis placed on developer design information and testing records. Rigorous, commercial-style development practices enforced without need for specialist knowledge. Moderate to high level of independent assurance. "White box" testing.
- EAL5 - Semiformal design and testing; rigorous development practices with moderate reliance on specialized engineering techniques.
- EAL6 - Semiformal verified design and testing; high assurance gained through specialized engineering techniques and rigorous development with strong configuration management, for use in high-risk applications where value of assets justifies the added cost.
- EAL7 - Formal verified design and testing; for use in extremely high risk applications. System design must lend itself to formal analysis.

Though they were intended for security planning, with some slight changes in language these levels could apply equally well to authentication planning. They operate cumulatively, with each successive level building on the assurance of the last. Within this hierarchy, only the least probing levels may be achieved independent of design knowledge. Testing techniques rise in cost as the knowledge background to perform them becomes more specialized.

ACKNOWLEDGEMENTS

This work was completed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-ENG-48.

REFERENCES

1. Joint DoD/DOE Information Barrier Working Group, *The Functional Requirements and Design Basis for Information Barriers*, May 1999.
2. Carnegie-Mellon University, *Capability Maturity Model for Software, Version 1.1*, CMU/SEI-93-TR-024, ESC-TR-93-177, February 1993.
3. International Organization for Standardization/International Electrotechnical Commision, *Guide for Information Technology--Software Lifecycle Processes, 1997,* ISO/IEC 12207.
4. Federal Information Processing Standards (FIPS) Publication 140-1, *Security Requirements for Cryptographic Modules*, 11 January 1994.
5. International Organization for Standardization/International Electrotechnical Commision, *The Common Criteria for Information Technology Security Evaluation, 1999* [ISO/IEC 15408/1999].

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551